THE QL REPORT

PUBLISHED BY CURRY COMPUTER
P.O. BOX 5607, GLENDALE, AZ 85312-5607
$15.00 FOR 12 ISSUES

OCTOBER 15, 1987

VOL. 3 NO. 10

We lead off this issue by apologising for jumping the gun about the battery clocks. We knew they were coming and would be here shortly after we went to press and "assumed" that they would work fine, coming from a reliable source.

As it turned out, that batch would not work at all. Everytime we set the clock, powered down, and turned the QL back on, the clock was not correct. It is also very difficult to install. The board plugs into the 8302 chip socket. The 8302 plugs into the board. The board will rub against a diode on the right side of the 8302 socket which will have to be "pushed" out of the way. You also will have to unplug the two ribbon connectors to the keyboard in order to get to the 8302 socket. THIS IS NOT AN EASY BOARD TO INSTALL!! Please be very careful.

As a sideline to the above, if any of you ever experience dead keys on your QL, the first thing to suspect, and rightly so, is the membrane keyboard. However, sometimes, instead of having to replace it, you might find that upon close examination of the two cables that plug into the PCB, there will be hairline cracks at the base which are causing the problem. If you "trim" a quarter inch or so off with a pair of sharp scissors and re-install them into the sockets, you might find this clears up everything fine.

FRACTURED FRACTALS
BY Marshall Stiles ★★★★

You have no doubt seen some computer generated landscapes or other objects. If you saw any of the Star Trek movies, you did see a computer representation of an object on the screen made up of repetative squares, triangles, circles or curving lines. Unless the geometric objects making up the surfaces are filled in or shaded, you can "see through" the grid like surfaces. These are many times referred to as fractal representations.

The program below demonstrates the basic principle of how a drawing of this type can be made to represent uneven surfaces. It does not generate a vast landscape using the repetative operations on a decreasing scale like a true fractal generator. What it does do, is to let you manipulate a rectangular looking object divided into four triangles. It decreases in scale toward the upper part of the screen to give the illusion of distance. If you think of the object as a flat parcel of land at sea level, you will get the idea. Keys "1" to "5" allow you to raise the five points of intersect in the drawing, one dot at a time, above sea level. "ESC" lets you quit, in case you want to start over.

You should notice several things. If you raise each point the same number of key strokes the object is the same, but x number of dots above sea level. Pressing the "5" key will raise the center point and will create a 3 demensional pyramid. Raising only the outer points (keys "1" to "4") will create an inverted pyramid.

To create an impressive fractal representation you would want a large number of these geometric shapes, smaller in size and connected to one another. This would be kind of like having a see through quilt made of shapes of relative size. Hopefully the following program will help you get a good picture of what is taking place in this type of display!

```
110    REMark    fractal1_bas,    Marshall    Stiles,    1987
120 PAPER 2: CLS : PAPER 0 : INK 7
130 initialise
140 drawline
150 keyscan
999 STOP
1000 DEFine PROCedure initialise
1010 llx = 1 : lly = 1 : REMark lower left
1020 lrx = 100 : lry = 1 : REMark lower right
1030 ulx = 25 : uly = 50 : REMark upper left
1040 urx = 75 : ury = 50 : REMark upper right
1050 Cx = 50 : Cy = 33 : REMark center point
1060 END DEFine initialise
2000 DEFine PROCedure drawline
2010 LINE llx,lly TO lrx,lry TO Cx,Cy TO llx,lly
2020 LINE llx,lly TO ulx,uly TO Cx,Cy
2030 LINE ulx,uly TO urx,ury TO Cx,Cy
2040 LINE urx,ury TO lrx,lry
2999 END DEFine drawline
3000 DEFine PROCedure keyscan
3010 REPeat loop1
3020 k$=INKEY$
3030 IF CODE(k$) = 53 : Cy = Cy + 1 : drawline
3040 IF CODE(k$) = 49 : lly = lly + 1 : drawline
3050 IF CODE(k$) = 50 : uly = uly + 1 : drawline
3060 IF CODE(k$) = 51 : ury = ury + 1 : drawline
3070 IF CODE(k$) = 52 : lry = lry + 1 : drawline
3080 IF CODE(k$) = 27 : EXIT loop1
3900 END REPeat loop1
3999 END DEFine keyscan
//
```

BASIC ARCHIVE

ARCHIVE, as everyone knows, is a very powerful database. Unfortunately, it is only as powerful as you are a programmer. Anyone can create a mailing list file, or a customer file or a stockfile. It is the ability to write procedures which utilize the information in the records that makes ARCHIVE one of the better databases on any computer.

We are going to start a new series in QLR called BASIC ARCHIVE which will attempt to show you how to write procedures first on a rather simplistic level and then on a more complex one. In this issue we are going to show you a procedure that will print mailing labels. We will start with one that prints one record at a time and then show you how to modify it so that you can print a selected subfile.

For single sheet mailing labels, you need 6 "lprint" statements so the labels always line up correctly. If you have a database file with the fields-- fname$, lname$, street$, city$, state$ and zip$, you could create a procedure called LABELS which would look like this:

```
proc labels
lprint fname$;" ";lname$
lprint street$
lprint city$;
lprint state$;" ";zip$
lprint
lprint
endproc
```

You could, if you wanted put the city and state fields on the same line. As long as you have 6 "lprint" statements, the labels will line up correctly.

With this procedure, you can, whenever a record is on the screen, type "labels" and a mailing label will be generated for that specific record. This is fine if you are only in need of a few labels. But, suppose, like us, you have a newsletter or want to send out a flyer to your customers or even birthday invitations to 25 or 50 or 500 people? To have to print the labels one at a time would be very time-consuming to say the least.

Most of you, we hope, know how to use the SELECT command from within ARCHIVE. If you want to create a subfile of people who are only from the state of California, you would type in : select state$="CA" and after awhile, you would have a file that only contained people who live in that state.

This is what we do with the QL Report. We have a numeric field named "issues." This is where we log in the number of issues you have left in your subscription. If you order for all of 1987 in January of 1987, we log you in for 12 issues. Everytime we generate a mailing label, it subtracts one from the issues field. So, when we print labels for all our subscribers in January, if we look at the "issues" field, it will then show the number "eleven." We have written the procedure so that a label is generated only if "issues>0."

The procedure for the above starts by selecting "issues>0." It then begins printing the entire subfile. How, most people do not need numeric calculations so below is a procedure to print "all" of anything:

```
proc printer
input "position";x
position x
while not eof()
lprint fname$(1);" ";lname$
lprint street$
lprint city$;" ";state$
lprint " ";zip$
lprint recnum()
lprint

next
endwhile
endproc
```

You will notice some variations between this procedure and the previous one. Every record you enter into ARCHIVE is assigned a record number. You can find out the current record number of a record on the screen by typing: "print recnum()" and the number will be given. As you add records, and depending on how the file is ordered or selected, this number may change.

However, once you have selected your subfile which you want to print, each record's number will stay the same for that session. Before printing, the procedure asks you for the record number you want to begin with. We do this because when you print over 50 labels at a time, if your labels bind or you run out of labels prematurely, or whatever, you can see what number record was the last one that was okay and then when you start the procedure again, you know exactly where to start from again. As you can see, one of the "lprint" lines does print the record number. If you do not want to do this, that's up to you.

The main difference over this procedure and the previous one is the use of the "while not eof()" line. This construct, is a very important database concept to grasp, and exceedingly powerful. Once you have selected your subfile, with this construct, the procedure systematically starts at the first file and continues doing the instructions (in this case the "lprint" statements) until the last record in the file. So now, instead of having to manually "run" the procedure to print each label, you merely type in the word "printer" (ie, the name of the procedure) and all the selected records will be printed.

Next month we will show how you can use the UPDATE command to very easily make changes on either a single record or a large group of records.

BITS AND PIECES

We have some items we want to mention. Brett Lidstone in Canada is having problems connecting a Hayes 300/1200 baud modem to his QL. He cannot get it to work either with the modapter or just with a straight cable. If anyone out there has been successful in hooking up a Hayes, write us and we'll forward the information to Brett.

We are going to start a small classified section for subscribers who want to try and sell used equipment they have or who are looking for a piece of used equipment to purchase. You must be a subscriber of QLR and the word count must not exceed twenty words INCLUDING your name and address. The cost is 25 cents per word.

NEW VERSION OF SIDEWINDER

A new version of SIDEWINDER has been released which will make it more compatible with more printers. The initial version was only intended for printers which were fully compatible with the Epson RX80. Version 0.2 is more tolerant of "sort of" compatible printers and it can with a few glances at your own printer manual, be user configured for your printer providing it makes a little pretence at being Epson compatible. If your printer is capable of using any 8 pin Epson screen dump, then SIDEWINDER should be configurable for your printer. If the dump turns out none proportional, you will be able to configure it to be proportional. The Epson RX80 has 6 different 8 pin graphics modes and your printer may not have as many as this. You can configure the program so only your valid modes are displayed in SIDEWINDER.

If you have a printer that uses <ESC "K"> for a bit image mode but instead of Epson's 480 dots per 8 inches your printer does 640, you can inform SW and it will act accordingly. To be compatible with SW's 3 densities on Y printer axis, it is necessary that your printer uses Epson's <ESC "3"> code for 1/216" linefeed. If you do not have this don't panic because the triple density mode is only preferable when doing quite small screen dumps of 2" or smaller. If you only have 2 graphic modes and cannot use SW's true 3 Y axis densities, you will still be happy with the program as there is nothing worth comparing with it.

Considering that you can dump both upright or sideways, full partial screen dumps, and any any of these at any size, SW is an extremely useful program. Even if you only use it to knock out 20 foot "Happy Birthday" type banners.

WORKING WITH PASCAL FILES

→ → → → → By: Mel MacKaron ← ← ← ← ← ← ← ←

A reader wrote this month with a problem that is certainly not unique to programmers who are trying to learn Pascal. His problem was how to do I/O. While I went over his sample code, I found a number of other basic errors, again of a type that are not uncommon to novices to this language. Following is his listing, a general commentary on the errors and how to avoid them, and a sample program which performs simple I/O.

```
program introduction(input,output);

  const
    Numb1 = 1;
    Numb2 = 2;
    Numb3 = 3;
    Numb4 = 4;
    Numb5 = 5;

  type
    Arecord = record
      Field1 : integer;
      Field2 : integer;
      Field3 : integer;
      Field4 : integer;
      Field5 : integer
    end;

  var
    Fields : record;
    end;
    F : file of Arecord;

  begin
    rewrite(F):
    put(F);
    writeln('Begin File I/O')
  end.
```

In the very first line of the program one key element has been omitted: the name of the textfile. INPUT and OUTPUT are standard Pascal files, but you need to address the I/O file as I have done in my example.

Second, the author of this sample has defined a lot of constants (CONST) which are never used. It's best to delete them. While the MetaComCo compiler (which he used) will let you get away with this, others, such as a VAX mainframe will flag these with a warning message. Of course, they take up needless space and, in a time and space-critical environment, can make a major difference.

. The coding for RECORDS is correct, but they are never used (and should, too, be omitted). In my sample, I have avoided them and opted for an array of integers, as this is a simpler data format for the novice.

There are a number of errors in the VAR section. First, Fields is declared as type Record. Rather, it should be declared as of type Arecord, which was defined in the TYPE section. Second, the author placed an END statement within the VAR section. This in itself would cause the program to crash. You should never use END in a VAR section. Only use it for loops, procedures, functions, record declarations, WITH and CASE statements, and, of course, at the end of your program.

. In the main part of the program, designated by the keyword BEGIN, he begins by (correctly) opening the file; however, he never attached a name to the file. In Super-
however, he never attached a name to the file. In Super-
BASIC, for example, you can use a construction such as

```
open #4,mdv2_testfile
```

The same applies here. Incidentally, you can only write to a FILE of TEXT in Pascal (with some few exceptions). Therefore, the declaration of F as a FILE of ARECORD in VAR won't work; you should declare (filename) of type TEXT. Then, use the following statement: rewrite(f,filename);. This will attach the name to the output file. Afterward you will only refer to 'f'. This is clearly (sic) shown in my sample program.

The statement PUT(F) on the next line doesn't tell the program what to PUT into file F. [Personally, I seldom use PUT and GET, as most Pascals support the use of WRITE and READ in these situations.]

Enter the program listing I have provided and try to trace out how it accomplishes its task. Run it. It works just fine. If you want another example of programming, look at the Pascal article in last month's QL Report. It, again, demonstrates I/O in Pascal, both to and from disk and to the printer.

```
program introduction(input,output,storage);

  (This program is a demonstration of saving and retrieving
   data from a Pascal program.)

  const
    devname = 'mdv _testfile';

  type

    fournum = array[1..4] of integer;  (work w/four numbers)

    outdev = packed array [1..13] of char; (holds filename)

  var

    i,
    value : integer;

    drive : char;

    data : fournumbers;

    IOdevice : outdev;

    storage : text;               (must declare file type)

  begin (main program)

    write('This is a program which prompts the user');
    writeln('to enter four');
    write('numbers.  It will then ask for the microdrive');
    writeln('to use, will');
    write('save the numbers, and will then retrieve them');
    writeln('from the');
    writeln('storage device and print them on the screen.');
    writeln;
    write('Before proceeding, be sure that a FORMATTED');
    writeln('cartridge is in');
    writeln('the drive you wish to use.');
    writeln;
```

```
for i := 1 to 4 do
  begin
    write('Enter value ',i:1,': ');     {Enter value 1:}
    readln(value);
    data[i] := value                    {temp. storage}
  end;                        {4 values have been entered}

writeln;

repeat
  write('Enter drive number (1 or 2): ');
  readln(drive)
until drive in ['1','2'];    {must select either 1 or 2}

IOdevice := devname;                   {assign file name}
IOdevice[4] := drive;          {4th char := drive number}
rewrite(storage,IOdevice);        {open file for writing}
 for i := 1 to 4 do
   writeln(storage,data[i]);         {write data to file}

 writeln;
 write('Data is now stored.  It will now be read from ');
 writeln(IOdevice);
 writeln('and printed to the screen.');
 writeln;
 reset(storage,IOdevice);      {open file for read only}

 for i := 1 to 4 do
   begin
     readln(storage,data[i]);       {read data from file}
     writeln('Value ',i:1,' is ',data[i]:3)
   end;

 writeln;
 writeln('End of demonstration')

end.  {introduction}
```

If you have any questions concerning this article, or if you would like to see further Pascal problems addressed in QLR, please write to Curry Computer. If your query requires a personal response, please be sure to enclose an SASE (with adequate postage for the reply).

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

SANDY NEWS

We talked with Arnie Gardiner at Sandy UK recently and he told us the 256k battery-backed RAM card is still not ready but should be "very shortly- in about a month." The board will plug into the expansion port and between plug-ins to the QL should be able to keep information in memory for about six weeks.

Sandy also will have ready shortly a PC68k board which will plug into your IBM and allow your IBM to run QL programs. The board is more or less the same that FUTURA will have and will retail in the U.K. for approximately £200.00 (roughly $ 325.00). We hope to have more information on this board by next issue.

We will close this issue with some examples of screen dumps from Sidewinder. Next month we will continue our series on procedures in Archive and hope to have more information on Miracle's Midi interface and Sandy's new products.

UNTIL NEXT MONTH.....

ENJOY YOUR QL!